Multiobjective Artificial Immune Algorithm for Flexible Job Shop Scheduling Problem

Zohreh Davarzani[1], Mohammad-R Akbarzadeh-T[2] and Nima Khairdoost[3]

[1]*Department of Computer Engineering, Beyhagh Institute of Higher Education, Sabzevar, Iran*
[2]*Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*
[3]*Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Isfahan, Iran*
Z_davarni@yahoo.com, akbarzade@ieee.org,nima_khairdoost@yahoo.com

Abstract

Flexible Job shop scheduling is very important in production management and combinatorial optimization. It is NP-hard problem and consists of two sub-problems: sequencing and assignment. Multiobjective Flexible Job-Shop Scheduling Problems (MFJSSP) is formulated as three-objective problem which minimizes completion time (makespan), critical machine workload and total work load of all machines. In this paper a Multiobjective Artificial Immune Algorithm (MAIA) for FJSSP is presented. The proposed algorithm increases the speed of convergence and diversity of population. Kacem and Bradimart data are used to evaluate the effectiveness of MAIA. The experimental results show a better performance in comparison to other approaches.

Keywords: *Flexible Job Shop, Artificial Immune Algorithm, PPS mutation, Hypermutation, Clonal Selection*

1. Introduction

Task scheduling is one of the most critical problems in the economic domains and planning of manufacturing processes. Most scheduling problems are complex and solve difficulty. Scheduling may be defined as assignment of resources to tasks in such a way that a predefined performance measure is optimized.

One of the most common scheduling problems is job shop scheduling problem (JSSP), where a set of independent jobs must be processed on a set of available machines. This problem is one of the important optimization problems because it is used in the most planning and managing of manufacturing processes. Each job is a sequence of operations, each operation requiring a predefined machine. The problem is how to sequence the operations on the machines (sequencing) so that a predefined performance measure is optimized.

The Flexible Job-Shop Scheduling problem (FJSSP) is a special kind of classical JSSP, where operations are allowed to be processed on a subset of the available machines. Thus, FJSSP is more difficult than the classical JSSP because it has routing problem in addition to sequencing problem. The problem is NP-hard and consists of two sub-problems that need to be solved simultaneously. These sub-problems are as follows:

Sequencing problem: This problem is to sequence the operations on the machines over time.

Routing problem: it is to assign each operation to a suitable machine among of all machines.

To date, many approaches have been used to solve FJSSP, such as Tabu Search (TS)[1, 4, 5], Artificial Immune Algorithm (AIA)[7, 8, 9, 10, 13, 20], Branch-and-Bound (B&B), Genetic Algorithm (GA)[7,19], Simulated Annealing (SA) and hybrid of these methods [12, 17]. These available methods can be classified into two main categories: hierarchical approaches and integrated approaches.

In hierarchical methods, the sequencing of operations on the machines and routing are treated separately. Hierarchical methods decompose the original problem to sub-problems in order to reduce its complexity. In 1993, Brandimarte [1] first used this approach for the FJSSP. He solved the routing problem using dispatching rules and then solved the sequencing problem by TS heuristic. Kacem et. al., [18] presented a GA to optimize by the assigned model which was generated by localization approach localization. Xia and Wu [3] proposed a hybrid algorithm for the multi-objective FJSSP. They used Particle Swarm Optimization (PSO) for the routing problem on the machines and SA algorithm for the sequencing problem.

In contrast, integrated approaches solve sequencing and routing problems simultaneously. They are shown to reach better results than hierarchical methods, but are more difficult to solve. In 1994, Hurink et. al., [4] presented a TS algorithm in which reassignment and rescheduling are considered simultaneously. Dauzere-Peres and Paulli proposed a TS heuristic based on neighborhood structure solving problem [5]. In 2002, Mastrolilli and Gambardella [6] improved Dauzere-Peres' TS approaches and proposed two neighborhood functions.

Pezzellaa et. al., [7] solved the FJSSP using Genetic algorithm with the objective of minimizing makespan (maximum completion time). They presented multi-type individual to enhance efficiency of their method for solving problem. Xia et al. [3] proposed hierarchical approach for solving MFJSSP. They used particle swarm optimization (PSO) for operations assignment on machines and simulated annealing (SA) algorithm for operations scheduling on each machine. Objectives were minimizing makespan (maximal completion time), the total workload of all machines and the workload of the critical machine. Thomalla [26] has proposed a discrete-time programming model for FJSSP to minimize total weighted tardiness of all jobs. Choi [27] has proposed a MILP model for the problem with sequence-dependent setups. They used local search algorithm and dispatching rule to obtain an upper bound on the makespan of a sub-problem.

Among the above approaches, Artificial Immune Algorithm (AIA) is a well-known meta-heuristics which have been used for many optimization problems. Many Immune Algorithms have been proposed for solving Scheduling problems [7, 8, 9, 13, 20]. Ong et. al., [10] proposed an AIA called ClonaFLEX which was based on the Clonal selection principle, to solve FJSP. Bagheri et. al., [13] have proposed an Artificial Immune Algorithm for solving problem to minimize maximum completion time. A novel approach multi-modal Immune Algorithm is proposed for finding optimal solutions to JSSP by Guan et al. Chueh[24]. Zhang and Cheng [11] have proposed a hybrid simulated annealing algorithm. Their algorithm is based on immune mechanism with the objective of minimizing total weighted tardiness. Engin and Döyen [25] proposed a method based on clonal selection principle and affinity maturation mechanism for solving the flow shop scheduling problems.

In this paper to highlight the significant features of the Immune Algorithm, a Moltiobjective Artificial Immune Algorithm (MAIA) based on an integrated approach is proposed to solve a stochastic multiobjective FJSSP. This problem is formulated as a three-objective problem which minimizes completion time (makespan), workload of critical machine (machine with maximum load) and total work load of all machines.

The remainder of this paper is organized as follows: problem definition is given in section 2. Section 3 introduces main concepts of MOPs. Section 4 describes Artificial Immune Algorithm. The proposed algorithm is given in section 5. The computational results are given in section 6.

2. Definition of Flexible job-shop Scheduling Problem

The problem is to execute $N= \{J_1 \dots J_N\}$ jobs on $U= \{M_1 \dots M_m\}$ machines. Each job J_i is a set of n_i operations $\{O_{i,1} \dots O_{i,ni}\}$. Each operation $O_{i,j}$ can be processed on any subset $U_{i,j} \subseteq U$ of available machines. Each job is completed when its operations are executed one by one in a given sequence.

FJSSP consists of the following two sub-problems that need to be solved simultaneously. The first subproblem is the routing problem which is defined as determining the suitable machine from among the available machines for each operation, and the second problem is the problem of sequencing where the sequence of the assignment of operations to machines is determined over a required time span.

We wish to solve these sub-problems simultaneously in order to achieve the objectives which are minimizing makespan (i.e., the maximum job completion time), workload of critical machine (the machine with the highest workload) and workload of all machines.

FJSSP is classified into two sub-problems of: Partial FJSSP (P-FJSSP), and total FJSSP (T-FJSSP). We have partial flexibility if there exists a proper subset $U_{i,j}$ of U, for at least one operation $O_{i,j}$, while we have $U_{i,j} = U$ for each operation $O_{i,j}$ in the case of total flexibility [7].

Assumptions of this paper are as follows [4]:

1. Every machine processes only one operation at a time.
2. Machines are independent from each other.
3. Jobs are independent from each other.
4. There are no precedence constraints among the operations of different jobs.
5. Every operation is processed on only one machine at a time.
6. All jobs and their operations are available initially.

Table 1 presents the notations of this model.

Table 1. The Notations of Conceptual Model

M	Number of machines
N	Number of jobs
$O_{i,j}$	jth operation of job i
$T_{i,j}$	Processing time of $O_{i,i}$
$F_{i,j}$	Finish time of $O_{i,i}$
C_i	Completion time of job i
n_i	Number of operation of job i
$y_{m,i,j}$	assigned machine to operation $O_{i,j}$
W_j	workload of machine j
W_{max}	Total workload of all machine
C_{max}	Maximum completion time of all job

According to the notations above, model of FJSSP model can be defined as follows:

$$F_{ij} \leq T_{i,j+1} \quad i = 1, \dots, n; j = 1, \dots, n_i \tag{1}$$

$$F_{ij} \leq C_{max} \quad i = 1, \dots, n; j = 1, \dots, n_i \tag{2}$$

$$y_{mij} \leq a_{mij} \quad m = 1, \dots, M \tag{3}$$

$$\sum_{m=1}^{M} y_{mij} = 1 \tag{4}$$

Constraint (1) indicates precedence constraints among the operations in each job so that operations can be executed when its precedence operation is executed. Constraint (2) defines the makespan (C_{max}) and Constraint (3) indicates that each operation can be assigned to just one machine from among the given machines. Eq (4) shows that only one machine from the available alternatives can be assigned to each operation. In this paper, the following criteria are to be minimized:

$$f_1 = \max(C_i) \quad i = 1, \dots, N \tag{5}$$

$$f_2 = \sum_{i=1}^{M} \mathrm{w_k} \tag{6}$$

$$f_3 = \max(W_k) \quad k = 1, \dots, M \tag{7}$$

Eqs (5-7) indicate makespan or maximal completion time of machines, total workload of the all machines and workload of critical machine. Table 2 shows data related to a sample problem. In this table, rows and columns represent operations and machines respectively. Symbol '∞'means that a machine cannot process the corresponding operation. In other words, it does not belong to the subset of compatible machines for that operation.

Table 2. A Problem with Size 3*3

Job		m_1	m_2	m_3
J_1	$O_{1,1}$	1	2	2
	$O_{1,2}$	2	2	∞
	$O_{1,3}$	3	3	2
J_2	$O_{2,1}$	4	3	2
	$O_{2,2}$	2	∞	2
	$O_{2,3}$	3	2	3
J_3	$O_{3,1}$	2	2	5
	$O_{3,2}$	∞	2	3

3. Multiobjective Optimization Problem

Definition1. MOP has a number of objective functions which are maximized or minimized. Without loss of generality, a multiobjective minimization problem with *m* decision variables (parameters) and *n* objectives is defined as follows:

$$\begin{aligned} &minimize \quad y = f(x) = \left(f_1(x), f_2(x), \dots, f_k(x)\right) \\ &subject\ to \quad c = c(x) = (c_1(x), c_2(x), \dots, c_n(x)) \\ &where \quad x = (x_1, \dots, x_m) \in X \\ &\qquad y = (y_1, \dots, y_m) \in Y \end{aligned} \tag{8}$$

Where x the decision vector with n components, X decision space, y decision vector and Y is called the objective space. $c(x)$ determines the set of feasible solutions [22].

Definition 2. (Pareto dominance). A solution $p \in X$ is said to Pareto-dominate another $q \in X$(written p $<_n$ q) if two equations Eq. (9), (10) are true:

$$\{f_i(p) \leq f_i(q) |\ \forall\, i \in \{1, \dots, n\}\} \quad \textit{and} \tag{9}$$

$$\{f_i(p) < f_i(q) | \exists\, j \in \{1, \dots, n\}\} \tag{10}$$

Definition 3 (Pareto-Optimal Set). For a MOP, The Pareto optimal set (PS) is defined as follows:

$$PS = \{x \in X \mid \nexists x^* \leq_n x\}$$

Definition 4. (pareto front). For a given MOP and the Pareto-optimal set, the pareto front is defined as follows:

$$PF = \{f(x) = f_1(x), \dots, , f_n(x) | x \epsilon\, ps\} \tag{11}$$

4. Artificial Immune Algorithm

The immune system is an adaptive, self organizing and distributed system. In addition, it is a complex functional system that defends the human body from foreign agents such as viruses or bacteria called pathogens. It categorizes all cells or molecules into two kinds within the body: First are those that belong to its own kind (self-cell) and the second are that have a foreign origin (non-self-cell) [23].

Patterns expressed on pathogens are called antigens. The immune system contains cells for recognizing them. These cells are called antibodies. The disease procedure involves the attack of an antigen and its proliferation within the human body. After the proliferation of the antigen, antibodies are randomly distributed throughout the immune system.

AIS has two important processes: Cloning and affinity maturation. The combination of them is known as the Clonal Selection Principle. This principle is used to explain how the immune system reacts to infection of antigens [2]. Moreover, this theory is one of methodologies in AIS for solving optimization problems and is a meta-heuristic which is developed based on such a system. Fig1 shows this principle.

When a pathogen invades the human body, a number of cells that recognize pathogens proliferate. These cells can be classified into two kinds: First are effecter cells and second are memory cells. The effecter cells secrete antibodies in large numbers and the memory cells have long life spans so as to act faster and more effectively in future exposures to the same or a similar pathogen [13]. During cellular reproduction, the cells suffer somatic mutations at high rates, together with a selective force; the cells with higher affinity to the invading pathogen differentiate into memory cells. This whole process of somatic mutation plus selection is known as affinity maturation [14].

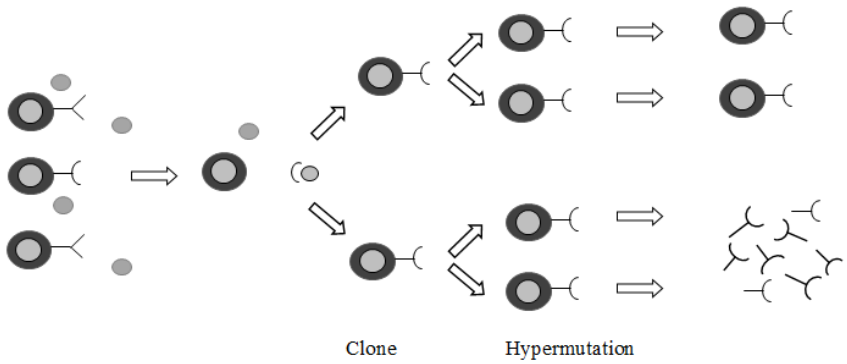

Figure 1. Clonal Selection Principle

5. Proposed Algorithm

In this section, we present a novel Multiobjective optimization algorithm called Multiobjective Artificial Immune Algorithm (MAIA). There are a set of antibodies Ab= {ab_1, ab_2,…,ab_N} in the population where *N* is the size of population. Each antibody represents a solution to the optimization problem. The proposed algorithm is given in Figure 2.

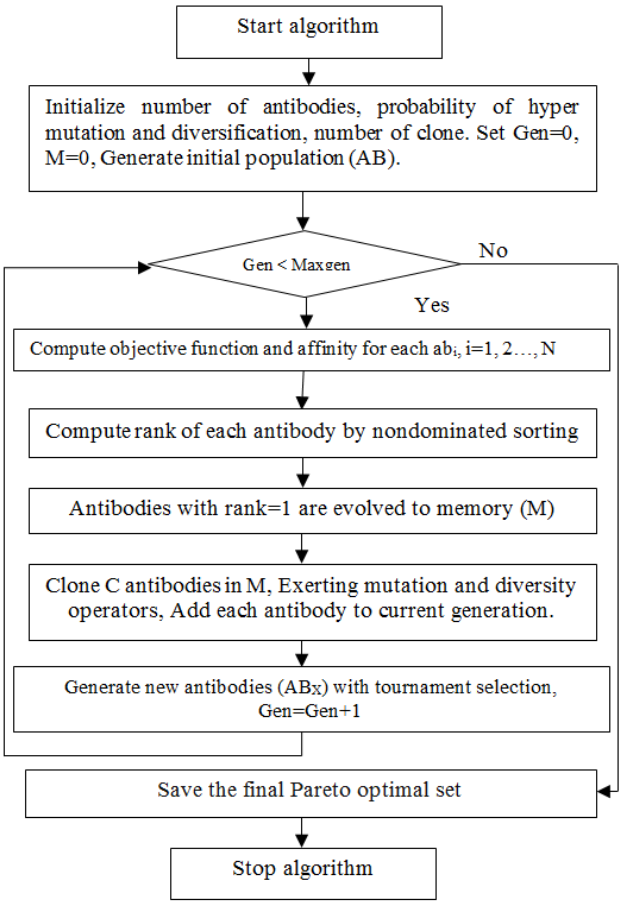

Figure 2. The Flowchart of the Proposed Algorithm

5.1. Antibody Representation

The main issue in applying AIA to FJSSP is to develop a method for finding an effective map between problem and solution generation. If this map is designed successfully, it is possible to find good solutions for an optimization problem.

In this paper, job sequencing provided by Kacem et. al., [16] is used to represent antibodies. Each antibody is a feasible solution that consists of many genes. Each gene represents an operation and consists of three parameters is formed by *(i, j, k)*, where

- *i* is the job that operation belongs to.
- *j* is the number of operations within job *i*.
- *k* is the machine assigned to that operation *j* of task *i*.

In this method the length of antibody is equal to $\sum_{i=1}^{N} n_i$ that *N* is the number of jobs and n_i is the number of operations of job *i*. For example consider the problem in Table 1. The string s={ (1,1,3),(2,1,2,),(1,2,2),(1,3,1),(3,1,1),(2,2,1),(3,2,2),(2,3,1)} is feasible solution from the solution space. The representation of antibody *s* is shown in Figure 3 and its gaunt chart is provided in Figure 4. In this example C_{max}, W_{maxv} and W_{td} are 10, 9 and 19 respectively.

(1,1,3)	(2,1,2)	(1,2,2)	(2,2,3)	(3,1,1)	(1,3,3)	(3,2,3)	(2,3,1)

Figure 3. Antibody Representation

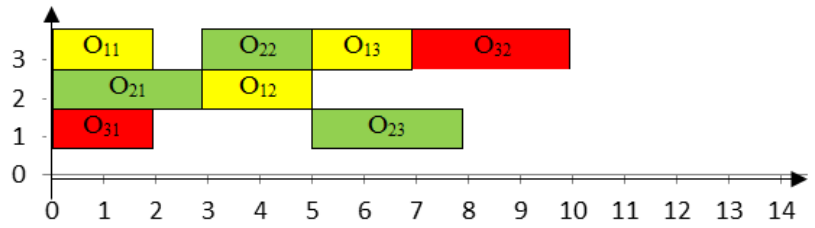

Figure 4. Gaunt Chart of Antibody s (C_{max} =10,W_{td}=19, W_{max}=9)

5.2. Non Dominated Sorting

In order to calculate the rank of each antibody *j*, a procedure is used in which the population is sorted into different nondomination levels. This procedure is proposed by Kalyanmoy [15] that is described in Figures 4 and 5. In this algorithm any antibody *i* has a *Counter* equal to zero, a *Rank*, and a *SD*. Also *Front (index)* is a set of antibodies in front index. For each antibody *i*, the *counter(i)* is the number of solutions which dominate the antibody *i*, and *SD(i)* is a set of solutions that the solution *i* dominates them.

In the first level, all antibodies have counters equal to zero. For each member *SD* of antibodies with *counter=0*, theire's counter is reduced by one. Then, any member with a zero counter is placed at the next front. The processed is continued until all levels are obtained.

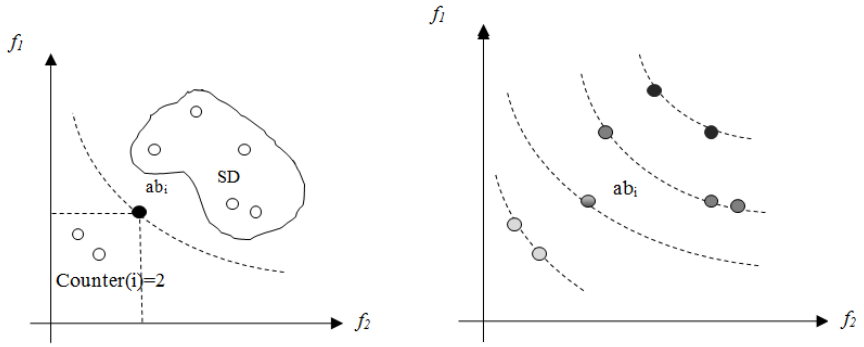

Figure 4. Nondominated Sorting Procedure

```
Input:   Antibodies  population,     Number of antibodies
SD(i)=∅, Counter(i) =0   ∀  i=,…, number_antibodies
Output: Nondomination Front
        For each antibody (a)
           Compare to all antibody (b)
             If    (a dominates b)
                    SD (a)=SD (a) ∪ b
             Else
             Counter (a)=Counter (a)+1
        If Counter (a) =0
          Rank(a)=1
          Front(1)=Front(1)  ∪ a
 index=1
for each antibody a in Front(index)
   for each antibody in SD (b)
     counter(b)=counter(b)-1
   if counter(b)==0
     Rank(b)= index+1
      Front(index)=Front(index) ∪ b
  index=index+1
```

Figure 5. Algorithm Fast non dominated Sorting

5.3. Mutation Operator

The mutation operator is the main operator in AIA. We use three kinds of mutation operator: Assignment mutation, Sequencing mutation and intelligent mutation.

Assignment operator changes the assignment of an operation to each machine. Sequencing operator changes the sequence of the operations. In applying the sequencing mutation, we must respect the precedence constrains among operations of the same job. In this paper, As in Kacem et. al., [21], the Precedence Preserving Shift mutation (PPS) operator is used.

Figure 6 shows assignment operator. In this antibody, positions 4 are selected randomly. Operation $O_{3,1}$ is in this position. This operator can be processed on machine $\{m_1, m_2, m_3\}$. m_3 is selected randomly and $O_{3,1}$ is assigned to it.

In Figure 7, an example is considered that shows the PPS mutation. In this figure, $O_{1,2}$ is chosen randomly among all of operation in the antibody. Feasible position for this operation, is 2, 3, 4 and 5. Position 5 is selected randomly and this operation is shift there.

In the intelligent mutation, we randomly choose an operation on the machine with the maximum workload (critical machine) and assign it to the machine with the minimum workload. For example in Figure 4, critical machine is m_3 and machine with minimum

workload is m_2. $O_{3,2}$ can process on m_2 with processing time 2. Therefore it is assigned to m_2 with processing time 1. Figure 8 shows this operator. By exerting this operator three objective functions improve.

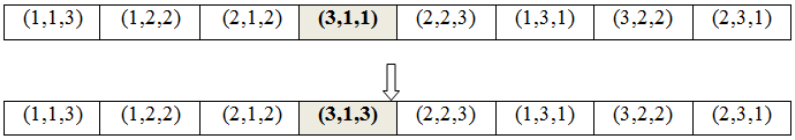

Figure 6. Assignment Operator

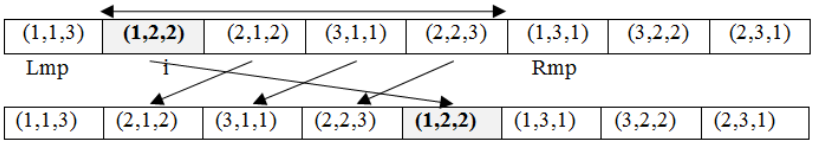

Figure 7. An Example of PPS Mutation (sequencing operator)

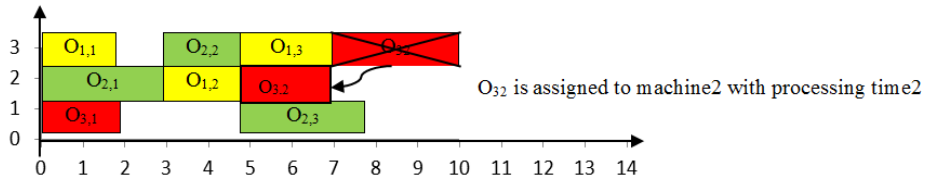

Figure 8. Intelligent Operator: C_{max}=8, W_{td}=8, W_{max}=18

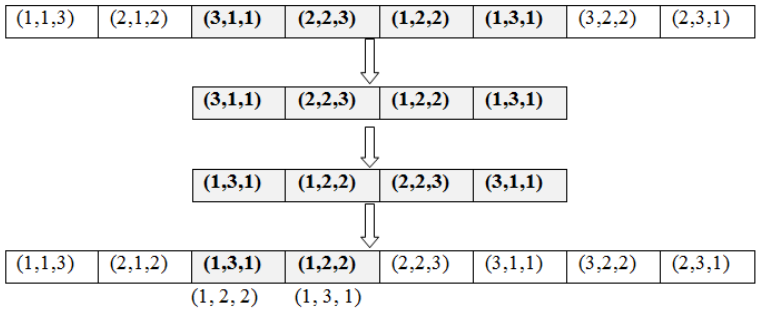

Figure 9. Antibody Inversion

5.4. Diversification Operator

Traditional immune system often has less diversity of the antibody. In this reason, it causes the search to be trapped in local optima. To avoid local optima, a diversification mechanism of Guan et. al., [24] is used in this paper that called antibody inversion. In this mechanism, a subset of consecutive subtasks is randomly chosen from antibody and inverse their location from front to rear. Inversion mechanism is shown in Figure 9.

6. Experimental Setup and Simulation

We implemented the proposed algorithm in MATLAB environment and run it on a PC with a 2.1 GHz CPU and 512 MB of RAM. The parameters for proposed algorithm are given in Table 3. Three data sets have been considered:

(1) Kacem data: This data set consists of three problems with different size: problem 8×8, problem 10×10 and problem 15×10 from Kacem et. al., [16]. Problem 8×8 has partial flexibility that consists of eight jobs with 27 operations which can be processed on 8 machines, Problem 10×10 has total flexibility that consists of 10 jobs with 30 operations which can be implemented on 10 machines, and Problem 15 ×10 has total flexibility that consists of 15 jobs with 56 operations which can be performed on 10 machines.

(2) BRdata: this data set consists of 10 problems from Brandimarte [1] that are randomly generated using a uniform distribution between two given limits. The number of jobs ranges from 10 to 20, the number of machines ranges from 4 to 15, the number of operations for each job ranges from 5 to 15, and the number of operations for all jobs ranges from 55 to 240. For example, problem 8*8 of Kacem data has been shown in Table 4. In this table, rows represent operations of jobs and machines are shown in the columns.

Table 3. Parameters of Proposed Algorithm

Parameter name	value
Population size	200
Number of generation	600
Mutation assignment probability	0.2
Mutation sequence probability	0.2
Diversification probability	0.7
Number of clone	100

Table 4. Problem 8×8 with 27 Operations (partial flexibility)

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
Job 1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
Job 2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
Job 3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
Job 4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
Job 5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
Job 6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
Job 7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
Job 8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

Tables 5, 6 and 7 indicate the best obtained solutions of 10 runs of the proposed algorithm on problems 8*8, 10*10 and 15*10 in comparison to other algorithms respectively. Five algorithms are used for comparison, which are Temporal Decomposition refers to Hammadi [16], Classic GA that is classical Genetic Algorithm, 'Approach by Localization' and 'AL & CGA' proposed by Kacem et al [18], AIA proposed by Bagheri [13] and Hybrid of PSO and SA proposed by Xia [3].

In these tables, first row indicates the name of algorithm, second, third and fourth rows present C_{max}, W_{td} and W_{max} respectively. In this table deviation criterion represented by *(Dev)* is employed. This criterion is defined as follows:

$$ev = \frac{f_{ap} - f_{best}}{f_{best}} \times 10 \quad (12)$$

where f_{best} the best objective function is obtained among all algorithms and f_{ap} is the objective function of the algorithm that we compare ours to. According to these tables, proposed algorithm can get more efficient results than other approaches but it gets the same result with hybrid of PSO and SA approach. Also scheduling results for problem 10×10 are shown in table 6. This table indicates our algorithm has better results than the approach by localization and temporal decomposition. In comparison to the hybrid of PSO and SA, Classical GA and the hybrid of AL and CGA approaches, proposed algorithm has better results in W_{td} and W_{max} and has worse C_{max} than them. Also, the best solution of problem 15×10 is shown in Table 7.

Table 5. Comparison of Results on Problem 8×8 with 27 Operations

	Temporal Decomposition[16]	Approach By localization[18]	AL+CGA [18]	PSO+SA [3]	Classical GA	MAIA
C_{max}	19	16	15	16	16	16
Dev%	26.66	6.06	0	6.06	6.06	6.06
W_{td}	91	75	79	73	77	73
Dev%	24.65	2.73	8.21	0	5.47	0
W_{max}	16	17	15	13	14	13
Dev%	23.07	30.76	15.38	0	7.69	0

Table 6. Comparison of Results on Problem 10×10 with 30 Operations

	Temporal Decomposition[16	Approach By localization[18]	AL+CGA [18]	PSO+SA [3]	Classical GA	MAIA
C_{max}	16	8	7	7	7	8
Dev%	128.56	14.28	0	0	0	14.28
W_{td}	59	46	45	44	53	41
Dev%	43.9	12.19	9.75	7.31	31.7	0
W_{max}	16	6	5	6	7	5
Dev%	220	20	0	20	40	0

Table 7. Comparison of Results on Problem 15×10 with 56 Operations

	AL+CGA[18]	PSO+SA[3]	MAIA
C_{max}	23	12	12
Dev%	91.66	0	0
W_{td}	95	91	89
Dev%	6.75	2.24	0
W_{max}	11	11	11
Dev%	0	0	0

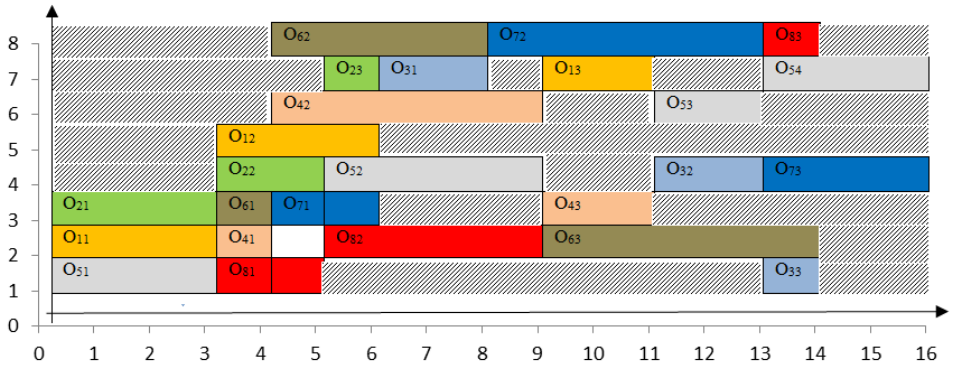

Figure 10. Optimization Solution of Problem 8×8 (W_{td}=73, W_{max} =13 and C_{max}=16)

Table 8. Results of MAIA on BRdata

		MAIS			AIA[13]		
Name	**size**	C_{max}	W_{td}	W_{max}	C_{max}	W_{td}	W_{max}
Mk01	10*6	45	36	162	40	36	171
Mk02	10*6	32	26	150	26	26	154
Mk03	15*8	216	204	1110	204	204	1207
Mk04	15*8	80	64	361	60	60	403
Mk05	15*4	180	173	683	173	173	686
Mk06	10*15	63	80	556	63	56	470
Mk07	20*5	140	140	750	140	140	695
Mk08	20*10	523	523	2627	523	523	2723
Mk09	20*10	320	306	2545	312	306	2591
Mk10	20*15	214	206	2100	214	206	2121

Figure 10 shows the Gaunt chart of problem 8×8. Second data set is BRdata. Table 8 shows the best results obtained by MAIA. In this table, problem names are presented in the first column; the second column is the size of problems. Third, fourth and fifth columns show the objective functions are obtained by the proposed algorithm and other columns present results are obtained by AIA suggested by bagheri [13]. Their approach was single objective and each objective function is obtained separately. As can be seen, the experimental results show that the proposed algorithm has better result in the most samples.

7. Conclusion

In this paper, we developed a Multiobjective Artificial Immune Algorithm base on Pareto optimality for the Flexible Job-shop Scheduling Problem (FJSSP). The consideration objective functions are minimizing maximum completion time of all jobs (makespan), critical machine workload and total workload of all machines. The proposed algorithm increases the speed of convergence and diversity of population. This algorithm was tested on two data sets of Kacem and Brandimart. In comparison to other algorithms, the proposed algorithm was shown to be more effective.

Reference

[1] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search", Annals of Operations Research, **(1993)**, pp. 157–83.
[2] G. L. Ada G. J. V. Nossal, "The clonal selection theory", Scientific American, **(1987)**, pp. 50-57.
[3] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", Journal of Computers & Industrial Engineering, **(2005)**, pp. 409–425.
[4] J. Hurink, B. Jurish and M. Thole, "Tabu search for the job shop scheduling problem with multi-purpose machines", **(1994)**, pp. 205–215.
[5] S. Dauz ére-P ér ès and J Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", Annals of Operations Research, **(1997)**, vol. 281–306.
[6] M. Mastrolilli, L. M. Gambardella, "Effective neighborhood functions for the flexible job shop problem", Journal of Scheduling, **(2002)**, pp. 3–20.
[7] F. Pezzellaa, G. Morganti and G. Ciaschetti, "A genetic algorithm for the Flexible Job-shop Scheduling Problem", journal of Computers & Operations Research, **(2008)**, pp. 3202 – 3212.
[8] S. Darmoul, H. Pierreval and S. H. Gabouj, "Scheduling using artificial immune system metaphors", IEEE International Conference on Service Systems and Service Management, **(2006)**, pp. 1150–1155.
[9] E. Hart and P. Ross, "An immune system approach to scheduling in changing environments", Proceedings of Genetic and Evolutionary Computation Conference, Orlando, Florida, **(1999)**, pp. 1559–1565.
[10] Z. X. Ong, J. C. Tay and C. K. Kwoh, "Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules", Applied: LNCS, **(2005)**, pp. 442-455.
[11] R. Zhang and C. Wu, "A hybrid immune simulated annealing algorithm for the job shop scheduling problem", Journal of Soft Computing, **(2010)**, pp. 79–89.
[12] G. Jinweia, G. Manzhan, C. Cuiwena and G. Xingshenga, "A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem", Computers & Operations Research, **(2010)**, pp. 927 – 937.
[13] A. Bagheri, M. Zandieh, I. Mahdavia and M. Yazdani, "An artificial immune algorithm for the flexible job-shop scheduling problem", Journal of Future Generation Computer Systems, **(2010)**, pp. 533-541.
[14] L. N. D. Castro and J. Timmis, "An artificial immune network for multimodal function optimization", Evolutionary computation, CEC'02, in: Proceedings of the 2002 Congress, **(2002)**, pp. 699-704.
[15] D. Kalyanmoy, "A Fast and Elitist Multiobjective Genetic Algorithm", NSGA-II, **(2002)**.
[16] I. Kacem, S. Hammadi and P. Borne, "Approach by localization and multi objective evolutionary optimization for flexible job-shop scheduling problems", IEEE Transactions on Systems, **(2002)**, pp. 1-13.
[17] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problem", Computers and Industrial Engineering, **(2005)**, pp. 409-425.
[18] I. Kacem, S. Hammadi and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic", Mathematics and Computers in Simulation, **(2002)**, pp. 245–276.
[19] H. Zhang and M. Gen, "Multistage-based genetic algorithm for flexible job-shop scheduling problem", Journal of Complexity International, vol. 11, **(2005)**, pp. 223-232.
[20] Z. X. Ong, J. C. Tay and C. K. Kwoh, "Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules", LNCS, **(2005)**, pp. 442-455.
[21] K. Lee, T. Yamakawa and K. M. Lee, "A genetic algorithm for general machine scheduling problems", International Journal of Knowledge-Based Electronic, **(1998)**.
[22] H. Zhi-Hua, "A multiobjective immune algorithm based on a multiple-affinity model", European Journal of Operational Research, **(2010)**, pp. 60–72.
[23] D. Dasgupta, "Special issue on artificial immune system", IEEE Transactions on Evolutionary Computation, **(2002)**, pp. 225-256.

[24] G. C. Luh and C. H. Chueh, "A multi-modal immune algorithm for the job-shop scheduling problem", Journal of Information Sciences, **(2009)**, pp. 1516–1532.
[25] O. Engin and A. Döyen, "A new approach to solve hybrid flow shop scheduling problems by artificial immune system", Future Generation Computer Systems, **(2003)**, pp. 1083–1095.
[26] C. S. Thomalla, "Job shop scheduling with alternative process plans", journal of Prod, **(2001)**, pp. 125–134.
[27] I. Choi and D. Choi, "A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups", Comput. Ind. Eng., **(2002)**, pp. 43–58.

Authors

Zohreh Davarzani received the BS and MS degrees in Computer engineering from Ferdowsi University of Mashhad, Iran in 2008 and 2011, respectively. Her research interests include fuzzy logic, soft computing and Quantum computing.

Mohammad-R Akbarzadeh-T received the PhD degrees in Electrical engineering from the Univercity of New Mexico in 1998. He currently is an professor in the Engineering Department of University of Mashhad. His research interests include Intelligent Control, Complex Systems, Computational Intelligence (Fuzzy Logic, Neural Networks, Genetic Algorithms, Reinforcement Learning), Multi-agent Systems, Mobile and Manipulator Robotics, Biomedical Engineering.

Nima Khairdoost received the BS and MS degrees in Computer engineering from Ferdowsi University of Mashhad and University of Isfahan, Iran in 2008 and 2011, respectively. His research interests include Image processing, soft computing, machine vision and pattern recognition as well as evolutionary algorithms.